

Лабораторная работа № 5 по курсу : Операционные системы

Выполнил студент группы М8О-201Б-21 Кварацхелия Александр.

Условие

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки 2-мя способами: Во время компиляции (на этапе «линковки»/linking) Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками В конечном итоге, в лабораторной работе необходимо получить следующие части: Динамические библиотеки, реализующие контракты, которые заданы вариантом; Тестовая программа (программа №1), которая использует одну из библиотек, используя знания полученные на этапе компиляции; Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты. Провести анализ двух типов использования библиотек.

Пользовательский ввод для обеих программ должен быть организован следующим образом: Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для программы №2). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»; «1 arg1 arg2 ... argN», где после «1» идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат её выполнения; «2 arg1 arg2 ... argM», где после «2» идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения.

Задание

Расчет интеграла функции $\sin(x)$ на отрезке $[A, B]$ с шагом e : (*FloatSinIntegral(float A, float B, float e)*). Методы: Подсчет интеграла методом прямоугольников, подсчет интеграла методом трапеций.

Расчет производной функции $float f'(x) = (f(A + deltaX) - f(A)) / deltaX$

Метод решения

Использовать утилиты компилятора *gcc* для динамической "линковки" библиотек.

Код программы

libLab5.h

```

typedef double(*TDerivative)(double, double);
typedef double(*TIntegral)(double, double, double);

double Derivative(double a, double deltaX);
double Integrate(double a, double b, double epsilon);
lib1.c

```

```

#include <math.h>

double Derivative(double a, double deltaX){
    return (cos(a + deltaX) - cos(a)) / deltaX;
}

double Integrate(double a, double b, double epsilon)
{
    int steps = fabs(b - a) / epsilon;
    double point = a;
    double result = 0;

    for (int i = 0; i < steps; ++i)
    {
        result += sin(point) * epsilon;
        point += epsilon;
    }

    return result;
}

```

lib2.c

```

#include <stdio.h>
#include <math.h>

double Integrate(double a, double b, double epsilon)
{
    int steps = fabs(b - a) / epsilon;
    double point = a;
    double result = 0;

    for (int i = 0; i < steps; ++i)
    {
        result += sin(point + epsilon / 2) * epsilon;
        point += epsilon;
    }
}

```

```

    }

    return result;
}

double Derivative(double a, double deltaX){
    return (cos(a + deltaX) - cos(a - deltaX)) / (2 * deltaX);
}

program1.c

#include <stdio.h>
#include <libLab5.h>

int main()
{
    printf("1_-_integrate ,_2_-_find_derivative ,_-1_-_quit\n");

    int command;

    while (scanf("%d", &command) != EOF)
    {
        if (command == 1)
        {
            printf("Pass_arguments:_\"a_b_epsilon\"\\n");

            double a, b, epsilon;
            scanf("%lf_%lf_%lf", &a, &b, &epsilon);

            printf("%lf\\n", Integrate(a, b, epsilon));
        }
        else if (command == 2)
        {
            printf("Pass_arguments:_\"a_deltaX\"\\n");

            double a, deltaX;
            scanf("%lf_%lf", &a, &deltaX);

            printf("%lf\\n", Derivative(a, deltaX));
        }
        else if (command == -1){
            break;
        }
    }
}

```

```

        else{
            return 0;
        }
    }

    return 0;
}

```

program2.c

```

#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#include "libLab5.h"

int main()
{
    char* libNames[] = { "./libdynamic1.so", "./libdynamic2.so" };
    int curLib = 0;
    int numOfLibs = 2;

    void* libCtrl;
    libCtrl = dlopen(libNames[curLib], RTLD_NOW);

    if (!libCtrl)
    {
        perror("Opening_dynamic_lib_error");
        exit(EXIT_FAILURE);
    }

    TDerivative derivative;
    TIntegral integral;

    #pragma GCC diagnostic push
    #pragma GCC diagnostic ignored "-Wpedantic"
        derivative = (TDerivative)dlsym(libCtrl, "Derivative");
        integral = (TIntegral)dlsym(libCtrl, "Integrate");
    #pragma GCC diagnostic pop

    printf("0__switch_library_(default__derivative),_1__find_derivative,_2

    int command;

```

```

while (scanf ("%d", &command) != EOF)
{
    if (command == 0)
    {
        curLib = (curLib + 1) % numOfLibs;

        if (dlclose(libCtrl) != 0)
        {
            perror("closing_dynamic_lib_error");
            exit(EXIT_FAILURE);
        }

        if (!(libCtrl = dlopen(libNames[curLib], RTLD_LAZY)))
        {
            perror("closing_dynamic_lib_error");
            exit(EXIT_FAILURE);
        }

        #pragma GCC diagnostic push
        #pragma GCC diagnostic ignored "-Wpedantic"
        derivative = (TDerivative)dlsym(libCtrl, "Derivative");
        integral = (TIntegral)dlsym(libCtrl, "Integrate");
        #pragma GCC diagnostic pop
    }
    else if (command == 1)
    {
        printf("Pass_arguments:␣\"a_deltaX\"␣\n");

        double a, deltaX;
        scanf("%lf%lf", &a, &deltaX);

        double result = (derivative)(a, deltaX);
        printf("%lf\n", result);
    }
    else if (command == 2)
    {
        printf("Pass_arguments:␣\"a_b_epsilon\"␣\n");

        double a, b, epsilon;
        scanf("%lf%lf%lf", &a, &b, &epsilon);

        double result = (integral)(a, b, epsilon);
    }
}

```

```

        printf("%lf\n", result);
    }
    else if (command == -1){
        break;
    }
    else{
        return 0;
    }
}

return 0;
}

```

Выводы

В процессе выполнения лабораторной работы были приобретены навыки практического применения создания, обработки и отслеживания состояния динамических библиотек.